IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Application of : **ZLOTNICK**

Serial No.: 09/917,719 : Group Art Unit: 2623

Filed : July 31, 2001 : Examiner: Jon C. Chang

For : SORTING IMAGES FOR IMPROVED DATA ENTRY PRODUCTIVITY

Honorable Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450

DECLARATION UNDER 37 CFR 1.131

Sir:

I, the undersigned, Aviad Zlotnick, hereby declare as follows:

1) I am the Applicant in the patent application identified above. I am the inventor of the subject matter described and claimed in claims 1-27 therein.

2) Prior to May 7, 2001, I reduced my invention to practice, as described and claimed in the subject application, in Israel, a WTO country. The invention was implemented in the form of software code in the C and TCL programming languages. This code was initially compiled and run in order to demonstrate a possible application of the invention in parking ticket processing for the City

of Chicago. Preparation of the present patent application began shortly thereafter.

3) As evidence of the reduction to practice of the present invention, I attach hereto in Exhibits A and B the software source code that I used to implement the invention (with the assistance of a TCL programmer on the IBM technical staff). A directory listing in Exhibit C (generated by the AIX operating system used in the IBM Haifa Research Laboratory) shows the dates on which the source code files, named ImgSort.c and ChicagoKeyIn.tcl, were stored on disk. The dates of the files, which are blacked out in the Exhibits, are prior to May 7, 2001.

4) Generally speaking, the software code in Exhibit A performs the functions of sorting images (of parking tickets) according to a measure of their similarity one to another. The software code in Exhibit B then presents the images in order to an operator and receives inputs from the operator specifying codes to be assigned to the images, wherein a single input action by the operator (pressing <Return>) indicates that the current image is to be assigned the same code as the preceding image. The following table shows the correspondence between the elements of the method claims in the present patent application and elements of the program code listed below:

US 09/917,719

Declaration under 37 C.F.R 1.131 by Aviad Zlotnick

Claim 1	**Exhibits A and B**
1. A method for data entry, comprising:	The code in Exhibit B performs the function of accepting information key-in (line 8).
receiving a plurality of images;	Exhibit A, lines 81-97, reads in images of parking tickets.
sorting the images into an order responsive to a measure of similarity between the images, so as to group similar images together in the order;	The routine "mat_score" in Exhibit A (lines 8-19) calculates similarity scores, while "mat_swap" (lines 22-50) tests the scores subject to swapping the order of images. The main body of the program in Exhibit A (lines 104-201) loops over all the images using these scores to sort the images by similarity.
presenting to an operator a first image among the images in the order, and receiving an input from the operator specifying a code to be assigned to the first image;	The first image is presented by the "create image" step in Exhibit B, line 222. The operator specifies the code to assign to the image at line 405.

presenting to the operator a second image, subsequent to the first image among the images in the order, along with the code specified by the operator for assignment to the first image; and	The second image is presented in the next pass through line 222 in Exhibit B. The last code entered is specified at line 405 for the operator to accept or change.
assigning the code to the second image responsive to a single input action by the operator, indicating that the second image is to be assigned the same code as the first image.	Line 412 in Exhibit B binds NextImage to the <Return> key, and line 164 shows that whatever is in the internal structures is copied to output.
Claim 2	
A method according to claim 1, wherein the plurality of the images comprise entries in fields in one or more form documents.	Parking tickets comprise entries in multiple fields that are filled in by the parking inspector. Examples include the inspector's badge number (Exhibit B, line 117) and the street (line 126).

Claim 3	
A method according to claim 2, wherein the one or more documents comprise multiple fields, and wherein receiving the plurality of the images comprises extracting the entries from a selected one of the fields in the documents.	The parking ticket entries are read out by a standard IBM document processing program and are then provided as input to the present code at lines 85-97 in Exhibit A.
Claim 4	
A method according to claim 1, wherein the images comprise alphanumeric characters, and wherein the code comprises alphanumeric codes input by the operator corresponding to the alphanumeric characters appearing in the first image.	Clearly, parking tickets comprise alphanumeric characters filled in by the parking inspector. In Exhibit B, line 405, an alphanumeric input is received from the operator corresponding to the contents of specified fields in the parking ticket.

Claim 5	
A method according to claim 4, wherein sorting the images comprises applying optical character recognition (OCR) to the images so as to associate OCR codes with the characters, and grouping the images according to the OCR codes.	OCR is typically based on extracting features from an image and then analyzing the features to identify the characters in the image. In the present case, the image features are analyzed by the img_to_graph function at lines 120 and 130 in Exhibit A. The feature graphs created in this manner are then used in calculating the similarity scores for grouping the images (line 135).
Claim 6	
A method according to claim 5, wherein grouping the images comprises finding at least an approximate match between a first string of the OCR codes associated with the characters in the first image and a second string of the OCR codes associated with the characters in the second image.	The similarity score (mat_score, lines 8-20 in Exhibit A) is applied to the features extracted from the tickets in line 120-123 or 130-133 of Exhibit A. If the similarity score for a new permutation is higher than the current similarity score (i.e., if there is an approximate match), the order of the tickets is swapped (lines 167-185).

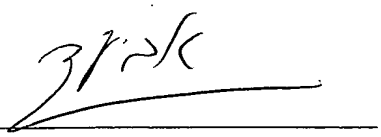
Claim 7	
A method according to claim 1, wherein the single input action comprises a single keystroke on a keyboard.	As noted above, line 412 in Exhibit B binds the single keystroke <Return> to the input action of indicating that the current image is to receive the same code as its predecessor.
Claim 8	
A method according to claim 1, wherein receiving the input from the operator specifying the code to be assigned to the first image comprises receiving a first input specifying a first code, and comprising, when the second image is not to be assigned the same code as the first image, receiving a second input from the operator specifying a second code to be assigned to the second image.	The "Results(street)" field in line 405 of Exhibit B serves both as output (showing the previous value) and as input (accepting a new value entered by the operator). A <Return> in this field accepts the previous value as the new value. Otherwise, the operator may key in the new value of the field.

Claim 9	
A method according to claim 8, and comprising presenting to the operator a third image, subsequent to the second image among the images in the order, along with the second code specified by the operator, and assigning the second code to the third image responsive to the single input action by the operator.	The procedure in Exhibit B (at line 390) loops over all the images in the input set. When the user inputs a code for a given image that is different from the preceding image, this code becomes the "Results(street)" that the operator can accept in the next image by pressing <Return>.

5) Claims 10-27 recite apparatus and a computer software product, with limitations similar to those of method claims 1-9. Based on the similarity of subject matter between the method, apparatus and software claims, it can similarly be demonstrated that I reduced to practice the entire invention recited in claims 10-27 prior to May 7, 2001.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and conjecture are thought to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the

validity of the application of any patent issued thereon.



Aviad Zlotnick, Citizen of Israel
Mizpe Netofa
D.N. Galil Takhton

Date:
Jan. 9th, '05

EXHIBIT A - C SOURCE CODE

File name : Imgsort.c

```
#include <stdio.h>
#include <cmacros.h>
#include <adt.h>
#include <mat.h>
#include <wordreco.h>
#include <img.h>

double mat_score(double **s, int n)
{
   double score;
   int    i, j;

   score = 0;

   for (i=0; i<n; ++i)
         for (j=i+1; j<n; ++j)
            score += s[i][j]/ABS(i-j);

   return (score);
}

double **mat_swap(double **s, int n, int k, int l)
{
   int i, j;
   double tmp, **new_s;

   mat_new(double, new_s, n, n);

   for (i=0; i<n; ++i)
         for (j=0; j<n; ++j)
            new_s[i][j] = s[i][j];

   s = new_s;

   for (i=0; i<n; ++i)
         {
         tmp = s[k][i];
         s[k][i] = s[l][i];
         s[l][i] = tmp;
         }


   for (i=0; i<n; ++i)
         {
         tmp = s[i][k];
         s[i][k] = s[i][l];
         s[i][l] = tmp;
         }

   return (s);
}
```

```
52
53  main(int argc, char **argv)
54      {
55          IMAGE *img1, *img2;
56          int       i, j, k, n_tickets, g, word_list, g1, g2, n, *p, m, prev;
57          long      s1, s2;
58          double    score, new_score, **s, **t;
59          int     il, nl, badge[1000], img_name[1000], unit[1000], start[1000];
60          char    ticket[1000][40], prev_in[200], prev_out[200];
61          FILE    *f;
62          int     u_in, u_out;
63
64          if (argc != 3)
65             printf (" **** usage: %s <input_list> <sorted_list> ****\n",
66                      argv[0]);
67
68          il = vpl_new();
69          nl = vpl_new();
70
71          u_in  = 0;
72          u_out = 0;
73
74          f = fopen(argv[1],"r");
75          if (f == NULL)
76             exit (1);
77
78          m = 0;
79          start[m] = 0;
80
81          for (n=0; fscanf(f,"%[^\n]\n",ticket[n]) > 0; ++n)
82             {
83             char str[200];
84
85             sscanf(ticket[n], "M%d.TIF%*d%d%d", img_name+n, unit+n, badge+n);
86
87             sprintf(str,"chicago/U%d/%d.img", unit[n], img_name[n]);
88             vpl_add(il, img_to_index(img_read(str),1));
89
90             if (unit[n] != prev)
91                    {
92                    prev = unit[n];
93                    start[++m] = n;
94                    }
95             }
96
97          fclose (f);
98
99          start[m] = n;
100         n_tickets = n;
101
102         f = fopen(argv[2],"w");
103
104         for (g=0; g<m; ++g)
```

```
105          {
106          n = start[g+1]-start[g];
107
108          AMALLOC("", p, INT, n);
109          for (i=0; i<n; ++i) p[i] = i;
110
111          if (n > 2)
112                {
113                mat_new(double, s, n, n);
114
115                for (i=0; i<n; ++i)
116                   {
117                   if(0)printf ("%4d)", start[g]+i);
118                   vpl_get(il, start[g]+i, (void **) &img1);
119
120                   g1 = img_to_graph((INDEXIMAGE *) img1,
121                                     0, img_n_rows(img1),
122                                     0, img_n_cols(img1));
123                   hwm_sign_word(g1,0);
124
125                   s[i][i] = 1.00;
126
127                   for (j=i+1; j<n; ++j)
128                         {
129                         vpl_get(il, start[g]+j, (void **) &img2);
130                         g2 = img_to_graph((INDEXIMAGE *) img2,
131                                           0, img_n_rows(img2),
132                                           0, img_n_cols(img2));
133                         hwm_sign_word(g2,0);
134
135                         score = match_words(g2, 0, g1, 0, 45, 30);
136                         s[i][j] = s[j][i] = SQUARE(score);
137
138                         if(0)printf (" %4.1f", score);
139
140                         grf_free(g2);
141                         wl_free(g2);
142                         }
143
144                   if(0)printf ("\n");
145
146                   grf_free(g1);
147                   wl_free(g1);
148                   if(0)img_free(img1);
149                   }
150
151                score = mat_score(s,n);
152                printf (" Mat score %8.3f\n", score);
153
154                for (k=0; k<n; ++k)
155                   {
156                   int count=0;
157
```

```
158                 for (i=0; i<n; ++i)
159                       for (j=0; j<n; ++j)
160                          {
161                          if (j != i)
162                                {
163                                t = mat_swap(s, n, i, j);
164
165                                new_score = mat_score(t, n);
166
167                                if (new_score > score)
168                                   {
169                                   int k;
170
171                                   k = p[i];
172                                   p[i] = p[j];
173                                   p[j] = k;
174
175                                   printf ("(%2d %2d)", i, j);
176                                   if(0)for (k=0; k<n; ++k) printf("%4d", p[k]);
177
178                                   score = new_score;
179                                   mat_free(s);
180                                   s = t;
181
182                                   printf (" %8.3f\n", score);
183
184                                   ++ count;
185                                   }
186                                else
187                                   mat_free(t);
188                                }
189                          }
190
191                   if (count == 0) break;
192                   }
193
194                mat_free(s);
195                }
196          else
197                for (i=0; i<n; ++i)
198                   {
199                   vpl_get(il, start[g]+i, (void **) &img1);
200                   if(0)img_free(img1);
201                   }
202
203          prev_in[0] = '\0';
204          prev_out[0] = '\0';
205
206          for (i=0; i<n; ++i)
207                {
208                char str[200];
209                FILE *b;
210
```

```
211                 sprintf(str, "chicago/U%d/%d.box",
212                         unit[start[g]+i], img_name[start[g]+i]);
213
214                 b = fopen(str,"r");
215                 if  (b != NULL)
216                    {
217                    fscanf(b, "%s", str);
218                    printf ("in  %s\n", str);
219                    fclose(b);
220                    }
221
222                 u_in += (strcmp(str,prev_in) == 0);
223                 strcpy(prev_in, str);
224                 }
225
226           for (i=0; i<n; ++i)
227                 {
228                 char str[200];
229                 FILE *b;
230
231                 fprintf(f, "%s\n", ticket[start[g]+p[i]]);
232
233                 sprintf(str, "chicago/U%d/%d.box",
234                         unit[start[g]+p[i]], img_name[start[g]+p[i]]);
235
236                 b = fopen(str,"r");
237                 if  (b != NULL)
238                    {
239                    fscanf(b, "%s", str);
240                    printf ("out %s\n", str);
241                    fclose(b);
242                    }
243
244                 u_out += (strcmp(str,prev_out) == 0);
245                 strcpy(prev_out, str);
246                 }
247
248           AFREE(p);
249           }
250
251     printf (" Repetitions out of %d: %d -> %d\n", start[m], u_in, u_out);
252     fclose(f);
253
254     exit (0);
255                                                 }
```

41261

EXHIBIT B - TCL SOURCE CODE

File name : ChicagoKeyIn.tcl

```
1   #! /opt/local/bin/wish -f
2
3   set DICT "/home/sommer/Parking/Dict"
4   set BADGE_DICT "/home/sommer/Parking/BadgeDict"
5   set img_dir "/home/sommer/Parking/BenchmarkStr"
6   #set img_dir "/home/sommer/Parking/BenchmarkGIF"
7
8   wm title . "Information KeyIn System"
9   wm geometry . +40+20
10
11  frame .dir -borderwidth 30
12  label .dir.label -text "Enter image directory: " \
13       -font -*-times-bold-r-*-*-20-*
14  entry .dir.entry -width 50 -relief sunken -font -*-times-medium-r-*-*-18-* \
15          -textvar img_dir
16  pack .dir.label -side left
17  pack .dir.entry -side left -fill x -expand true
18  pack .dir
19
20  focus .dir.entry
21
22  button .quit -text Quit -command exit
23  button .run -text Run  -command "StartImages .frame1"
24  pack .quit .run -side right -expand true -fill both
25
26  frame .frame1
27
28  bind .dir.entry <Return> "StartImages .frame1 "
29
30  set Results(unit) ""
31  set Results(badge) ""
32  set Results(street) ""
33  set Results(ticket) ""
34
35  set last_img ""
36  set list_type "UNIT"
37  set last_str_len 100
38  set last_list_len 100
39
40  foreach f [glob -nocomplain $BADGE_DICT/*.str] {
41    file delete -force $f
42  }
43
44  global res_fId
45  set res_fId [open KeyInData.res w]
46
47  proc ShowImageList {w} {
48    catch {destroy $w}
49    toplevel $w
50    wm geometry $w +800+200
51    wm title $w "Image Analysis"
52
53    global img_dir
```

```
54
55      global img_index
56      global img_list
57      global img_name
58      global img_id
59      set img_index 0
60      set img_name " "
61      set img_id " "
62
63      frame $w.frame1
64      frame $w.frame2
65      frame $w.frame1.f_img
66
67      frame $w.bot -borderwidth 10
68      button $w.bot.quit -text Quit -command "destroy $w"
69      button $w.bot.show -text Show -command "ShowImageInfo1 $w.frame1"
70      button $w.bot.next -text Next -command "NextImage $w.frame1 $w.comment.label"
71      pack $w.bot -side bottom -fill x  -expand true
72      pack $w.bot.quit -side left  -fill x -expand true
73      pack $w.bot.show -side left  -fill x -expand true
74      pack $w.bot.next -side left  -fill x -expand true
75
76      frame $w.f1
77      pack $w.f1
78      frame $w.f1.list -borderwidth 30
79      pack $w.f1.list -side right -expand yes -fill y
80      label $w.f1.list.header -text "Image List:  "
81      scrollbar $w.f1.list.yscroll -relief sunken \
82              -command "$w.f1.list.imgs yview"
83      scrollbar $w.f1.list.xscroll -relief sunken -orient horizontal \
84              -command "$w.f1.list.imgs xview"
85      listbox $w.f1.list.imgs -yscroll "$w.f1.list.yscroll set" \
86            -xscroll "$w.f1.list.xscroll set" \
87            -relief sunken -setgrid 1 -width 30 -height 10
88      pack $w.f1.list.header -anchor nw
89      pack $w.f1.list.yscroll -side right -fill y
90      pack $w.f1.list.xscroll -side bottom -fill x
91      pack $w.f1.list.imgs -side left -fill both -expand true
92
93      set img_list [exec cat img.lst]
94      $w.f1.list.imgs delete 0 end
95      foreach img_name $img_list {
96        $w.f1.list.imgs insert end $img_name
97      }
98
99      frame $w.comment -borderwidth 30
100     label $w.comment.label -text "Image Name: $img_id" \
101             -font -*-times-bold-r-*-*-20-*
102     pack $w.comment.label -side top -anchor w
103     pack $w.comment -side bottom -fill x
104
105     bind $w.f1.list.imgs <Button-1> \
106           "UpdateImageName $w.f1.list.imgs $w.comment.label %y"
```

```
107     bind $w.fl.list.imgs <Double-1> "ShowImage $w.frame1 \[selection get\]"
108
109   }
110
111   proc UpdateBadgeDict {} {
112     global BADGE_DICT
113     global Results
114
115     set strs_list ""
116
117     if {[file exists $BADGE_DICT/$Results(badge).str]} {
118       set fId [open $BADGE_DICT/$Results(badge).str r]
119       while {[gets $fId line] >= 0} {
120         set line_list [split $line ,]
121         lappend strs_list [string trimright [lindex $line_list 1] " "]
122       }
123       close $fId
124     }
125
126     set found [lsearch -exact $strs_list $Results(street)]
127     if {$found == -1} {
128       lappend strs_list $Results(street)
129       set new_strs_list [lsort $strs_list]
130       set fId [open $BADGE_DICT/$Results(badge).str w]
131       foreach str $new_strs_list {
132   #       puts -nonewline $fId $Results(unit)
133   #       puts -nonewline $fId " "
134         puts $fId $Results(unit),$str
135       }
136       close $fId
137     }
138   }
139
140   proc UpdateImageId { } {
141     global img_name
142     global img_id
143
144     puts $img_name
145     set str_start 0
146     set str_end [string first "." $img_name]
147     set str_end [expr $str_end - 1]
148     set f_name [string range $img_name $str_start $str_end]
149     set str_start [string last "/" $f_name]
150     set str_start [expr $str_start + 2]
151     set img_id [string range $f_name $str_start end]
152   }
153
154   proc NextImage {w} {
155     global img_index
156     global img_list
157     global img_name
158     global img_id
159     global Results
```

```
160     global ImageData
161     global last_img
162     global res_fId
163
164     puts $res_fId $Results(ticket),$Results(street)
165     UpdateBadgeDict
166
167     incr img_index
168     if {$img_index >= [llength $img_list]} {
169       set img_id " "
170       $w.imgId.imgnum configure -text "   Image Id: $img_id "
171       return
172     }
173     set last_img $img_name
174     set img_name [lindex $img_list $img_index]
175     UpdateImageId
176     set Results(ticket) $ImageData($img_name,TicketNbr)
177     set Results(unit) $ImageData($img_name,Unit)
178     set Results(badge) [format "%05s" $ImageData($img_name,BadgeNbr)]
179     $w.imgId.imgnum configure -text "   Image Id: $img_id "
180     $w.imgId.imgind configure -text "   Image Index: $img_index "
181     $w.imgId.ticket configure -text "Ticket No.: $Results(ticket) "
182     $w.imgId.unit configure -text "UNIT: $Results(unit)  "
183     $w.imgId.id configure -text "Badge No.: $Results(badge) "
184
185     ShowImage $w $img_name
186   }
187
188   proc UpdateImageName {w w1 y} {
189     global img_name
190     global img_index
191     global img_list
192     global img_id
193
194     set img_name [$w get [$w nearest $y]]
195     UpdateImageId
196     $w1 configure -text "Image Name: $img_id"
197
198     set img_index [lsearch -exact $img_list $img_name]
199
200     return "$img_name"
201   }
202
203   proc ShowImageInfo1 {w} {
204     global img_name
205
206     ShowImage $w $img_name
207   }
208
209   proc ShowImage {w i_name} {
210     global img_dir
211
212     set str_end [string first "." $i_name]
```

```
213       set str_end [expr $str_end - 1]
214       set f_name [string range $i_name 0 $str_end]
215       image create photo image1 -file $img_dir/$f_name.gif -palette 256
216       set img_w [image width image1]
217       set img_h [image height image1]
218
219       $w.c configure -width $img_w -height $img_h
220
221       $w.c delete image
222       $w.c create image 0 0 -image image1 -anchor nw
223
224       ShowList $w.keyin
225
226     }
227
228     proc AutoStringCont {n_list list_len} {
229       global Results
230
231       set cont_flag 1
232       set first_item [lindex $n_list 0]
233       set ind [string length $Results(street)]
234
235       while {$cont_flag == 1} {
236         set cont_char [string index $first_item $ind]
237         set total 1
238
239         for {set i 1} {$i < $list_len} {incr i} {
240           set cur_item [lindex $n_list $i]
241           if {[string index $cur_item $ind] == $cont_char} {
242             incr total
243           }
244         }
245
246         if {$total == $list_len} {
247           append Results(street) $cont_char
248           incr ind
249         } else {
250           set cont_flag 0
251         }
252       }
253     }
254
255
256     proc ShowList {w} {
257       global DICT
258       global BADGE_DICT
259       global Results
260
261       global list_type
262       global strs_list
263       global last_str_len
264       global last_list_len
265
```

```
266     $w.lst.strs delete 0 end
267     set strs_list ""
268     if {[file exists $BADGE_DICT/$Results(badge).str]} {
269       set list_type "BADGE"
270       set fId [open $BADGE_DICT/$Results(badge).str r]
271       while {[gets $fId line] >= 0} {
272         set line_list [split $line ,]
273             lappend strs_list [string trimright [lindex $line_list 1] " "]
274       }
275       close $fId
276       set strs_list_len [llength $strs_list]
277       for {set i 0} {$i < $strs_list_len} {incr i} {
278         $w.lst.strs insert end [lindex $strs_list $i]
279       }
280       set last_list_len $strs_list_len
281       if {$strs_list_len == 1} {
282         set last_str_len [string length $Results(street)]
283         set Results(street) [lindex $strs_list 0]
284       } else {
285   #       AutoStringCont $strs_list $strs_list_len
286         $w.str.entry icursor end
287       }
288     } else {
289       if {[file exists $DICT/$Results(unit).str]} {
290         set list_type "UNIT"
291         set fId [open $DICT/$Results(unit).str r]
292         while {[gets $fId line] >= 0} {
293           set line_list [split $line ,]
294           lappend strs_list [string trimright [lindex $line_list 1] " "]
295         }
296         close $fId
297         set strs_list_len [llength $strs_list]
298         for {set i 0} {$i < $strs_list_len} {incr i} {
299           $w.lst.strs insert end [lindex $strs_list $i]
300         }
301         set last_list_len $strs_list_len
302         if {$strs_list_len == 1} {
303           set last_str_len [string length $Results(street)]
304           set Results(street) [lindex $strs_list 0]
305         } else {
306   #         AutoStringCont $strs_list $strs_list_len
307           $w.str.entry icursor end
308         }
309       }
310     }
311
312     focus $w.str.entry
313     $w.str.entry icursor end
314
315   }
316
317   proc StartImages {w} {
318     catch {destroy $w}
```

```
319     toplevel $w
320     wm geometry $w +100+100
321     wm title $w "Image"
322
323     global Results
324     global ImageData
325
326     global img_dir
327     global img_index
328     global img_list
329     global img_name
330     global img_id
331     global last_img
332
333     set img_index 0
334     set img_name " "
335     set img_id " "
336
337     set img_list ""
338
339     set fId [open KeyInData.lst r]
340     while {[gets $fId line] >= 0} {
341       set img_name [lindex $line 0]
342       set ImageData($img_name,TicketNbr) [lindex $line 1]
343       set ImageData($img_name,Unit) [lindex $line 2]
344       set ImageData($img_name,BadgeNbr) [lindex $line 3]
345       lappend img_list $img_name
346     }
347     close $fId
348
349     set img_name [lindex $img_list $img_index]
350     set last_img $img_name
351     UpdateImageId
352     set Results(ticket) $ImageData($img_name,TicketNbr)
353     set Results(unit) $ImageData($img_name,Unit)
354     set Results(badge) $ImageData($img_name,BadgeNbr)
355
356     set str_end [string first "." $img_name]
357     set str_end [expr $str_end - 1]
358     set f_name [string range $img_name 0 $str_end]
359     image create photo image1 -file $img_dir/$f_name.gif -palette 256
360     set img_w [image width image1]
361     set img_h [image height image1]
362
363     canvas $w.c -width $img_w -height $img_h
364     pack $w.c
365
366     $w.c delete image
367     $w.c create image 0 0 -image image1 -anchor nw
368
369     frame $w.imgId -borderwidth 10
370     label $w.imgId.imgnum -text "   Image Id: $img_id " \
371                 -font -*-times-bold-r-*-*-20-*
```

```
372     label $w.imgId.imgind -text "   Image Index: $img_index " \
373                   -font -*-times-bold-r-*-*-20-*
374     label $w.imgId.ticket -text "Ticket No.: $Results(ticket) " \
375                   -font -*-times-bold-r-*-*-20-*
376     label $w.imgId.unit -text "UNIT: $Results(unit)  " \
377                  -font -*-times-bold-r-*-*-20-*
378     label $w.imgId.id -text "Badge No.: $Results(badge) " \
379                 -font -*-times-bold-r-*-*-20-*
380
381     pack $w.imgId.imgnum -side left
382     pack $w.imgId.imgind -side left
383     pack $w.imgId.ticket -side left
384     pack $w.imgId.unit -side left
385     pack $w.imgId.id -side left
386
387     frame $w.bot -borderwidth 10
388     button $w.bot.quit -text Quit -command "destroy $w"
389     button $w.bot.clear -text Clear -command "ClearStreet"
390     button $w.bot.next -text Next -command "NextImage $w"
391
392     pack $w.bot -side bottom -fill x  -expand true
393     pack $w.bot.quit -side left  -fill x -expand true
394     pack $w.bot.clear -side left  -fill x -expand true
395     pack $w.bot.next -side left  -fill x -expand true
396     pack $w.bot -side bottom
397
398     pack $w.imgId -side bottom
399
400     frame $w.keyin -borderwidth 10
401
402     frame $w.keyin.str -borderwidth 10
403     label $w.keyin.str.label -text "Street:    " -font -*-times-bold-r-*-*-20-*
404     entry $w.keyin.str.entry -width 20 -relief sunken \
405          -font -*-times-medium-r-*-*-18-* -textvar Results(street)
406     pack $w.keyin.str.label -side left -anchor n
407     pack $w.keyin.str.entry -side left -anchor n
408   #  pack $w.keyin.str.entry -side left -anchor nw -fill x -expand true
409     pack $w.keyin.str -side left -anchor n -expand true -in $w.keyin
410
411     bind $w.keyin.str.entry <KeyPress> "UpdateList $w.keyin %A; break"
412     bind $w.keyin.str.entry <Return> "NextImage $w"
413     bind $w.keyin.str.entry <Escape> "ClearStreet"
414     bind $w.keyin.str.entry <BackSpace> "DeleteChar $w.keyin; break"
415
416     label $w.keyin.dummy -text "      "
417   #  pack $w.keyin.str -side left -expand true
418     frame $w.keyin.lst
419   #  pack $w.keyin.str.lst -side right -expand yes -fill y
420     pack $w.keyin.lst -side right -anchor e
421     pack $w.keyin.dummy -side left -expand true -fill both
422     scrollbar $w.keyin.lst.yscroll -relief sunken \
423           -command "$w.keyin.lst.strs yview"
424     scrollbar $w.keyin.lst.xscroll -relief sunken -orient horizontal \
```

```
425             -command "$w.keyin.lst.strs xview"
426       listbox $w.keyin.lst.strs -yscroll "$w.keyin.lst.yscroll set" \
427             -xscroll "$w.keyin.lst.xscroll set" \
428             -relief sunken -setgrid 1 -width 15 -height 10
429       pack $w.keyin.lst.yscroll -side right -fill y -in $w.keyin.lst
430       pack $w.keyin.lst.xscroll -side bottom -fill x -in $w.keyin.lst
431       pack $w.keyin.lst.strs -side left -in $w.keyin.lst
432
433       pack $w.keyin -anchor w -fill x -expand true
434
435       ShowList $w.keyin
436
437
438    }
439
440    proc UpdateList {w char} {
441       global DICT
442       global Results
443       global strs_list
444       global img_name
445       global last_img
446       global list_type
447       global last_str_len
448       global last_list_len
449    #  global chars
450
451    #  if {[lsearch -exact $chars $char] == -1} {
452    #     return
453    #  }
454
455       if {[string compare $img_name $last_img] == 0} {
456          if {$last_list_len == 1} {
457                set Results(street) [string range $Results(street) 0 [expr $last_str_len-1]]
458          } else {
459             set Results(street) [$w.str.entry get]
460          }
461       } else {
462          set Results(street) ""
463          set last_img $img_name
464       }
465       set addr_str $Results(street)
466       append addr_str $char
467       set str [string toupper $addr_str]
468    #  set reg [string toupper $addr(reg)]
469       set Results(street) $str
470       update idletasks
471
472       set new_strs_list ""
473       set len [llength $strs_list]
474       for {set i 0} {$i < $len} {incr i} {
475          set cur_name [lindex $strs_list $i]
476          if {[string match $Results(street)* $cur_name]} {
477             lappend new_strs_list $cur_name
```

```
478       }
479     }
480     set strs_list_len [llength $new_strs_list]
481
482     if {$strs_list_len == 0} {
483       if {$list_type == "BADGE"} {
484         if {[file exists $DICT/$Results(unit).str]} {
485           set list_type "UNIT"
486           set fId [open $DICT/$Results(unit).str r]
487           while {[gets $fId line] >= 0} {
488             set line_list [split $line ,]
489                 lappend strs_list [string trimright [lindex $line_list 1] " "]
490           }
491           close $fId
492         }
493         set new_strs_list ""
494         set len [llength $strs_list]
495         for {set i 0} {$i < $len} {incr i} {
496           set cur_name [lindex $strs_list $i]
497           if {[string match $Results(street)* $cur_name]} {
498             lappend new_strs_list $cur_name
499           }
500         }
501         set strs_list_len [llength $new_strs_list]
502       }
503     }
504
505     set last_list_len $strs_list_len
506
507     $w.lst.strs delete 0 end
508     for {set i 0} {$i < $strs_list_len} {incr i} {
509       $w.lst.strs insert end [lindex $new_strs_list $i]
510     }
511     if {$strs_list_len == 1} {
512        set last_str_len [string length $Results(street)]
513       set Results(street) [lindex $new_strs_list 0]
514     }
515   #  if {$strs_list_len > 0} {
516   #    if {$strs_list_len == 1} {
517   #      set Results(street) [lindex $new_strs_list 0]
518   #    } else {
519   #      AutoStringCont $new_strs_list $strs_list_len
520   #    }
521   #  }
522     $w.str.entry icursor end
523   }
524
525   proc ClearStreet {} {
526     global Results
527
528     set Results(street) ""
529   }
530
```

```
531   proc DeleteChar {w} {
532     global Results
533     global strs_list
534     global img_name
535     global last_img
536   #  global chars
537
538   #  if {[lsearch -exact $chars $char] == -1} {
539   #    return
540   #  }
541
542     set str_len [string length $Results(street)]
543     set new_len [expr $str_len - 2]
544     set addr_str [string range $Results(street) 0 $new_len]
545   #  set addr_str $Results(street)
546     set str [string toupper $addr_str]
547   #  set reg [string toupper $addr(reg)]
548     set Results(street) $str
549     update idletasks
550     set new_strs_list ""
551     set len [llength $strs_list]
552     for {set i 0} {$i < $len} {incr i} {
553       set cur_name [lindex $strs_list $i]
554       if {[string match $Results(street)* $cur_name]} {
555         lappend new_strs_list $cur_name
556       }
557     }
558     $w.lst.strs delete 0 end
559     set strs_list_len [llength $new_strs_list]
560     for {set i 0} {$i < $strs_list_len} {incr i} {
561       $w.lst.strs insert end [lindex $new_strs_list $i]
562     }
563     if {$strs_list_len == 1} {
564       set Results(street) [lindex $new_strs_list 0]
565     }
566   #  if {$strs_list_len > 0} {
567   #    if {$strs_list_len == 1} {
568   #      set Results(street) [lindex $new_strs_list 0]
569   #    } else {
570   #      AutoStringCont $new_strs_list $strs_list_len
571   #    }
572   #  }
573     $w.str.entry icursor end
574
575   }
```

EXHIBIT C - AIX DIRECTORY LISTINGS

```
-rw-r--r--  1 aviad    None     81328 Nov 22  2000 #output#
-rw-r--r--  1 aviad    None     85219 Aug  9  2001 0
-rw-r--r--  1 aviad    None       490 Apr 20  1994 013.box
-rw-r--r--  1 aviad    None        72 Jan 27  1994 013.lst
-rw-r--r--  1 aviad    None      7719 Apr 20  1994 013.mmr
-rw-r--r--  1 aviad    None      8119 Apr 20  1994 014.mmr
-rw-r--r--  1 aviad    None        21 Jan 27  1994 016.box
-rw-r--r--  1 aviad    None     85363 Aug  9  2001 1
-rw-r--r--  1 aviad    None     16768 Nov 28  2000 KeyInData.lst
-rw-r--r--  1 aviad    None       544 Nov 28  2000 M.awk
-rw-r--r--  1 aviad    None       455 Nov 28  2000 MakeKeyInList.awk
-rw-r--r--  1 aviad    None      1156 Nov 21  2000 Makefile
-rw-r--r--  1 aviad    None      3061 Nov 22  2000 U.lst
-rw-r--r--  1 aviad    None     23148 Nov 22  2000 Unit.lst
-rw-r--r--  1 aviad    None     38202 Jan 30  2001 ZF.opt
-rw-r--r--  1 aviad    None       728 Nov 20  2000 a.awk
-rwxr-xr-x  1 aviad    None       753 Nov 16  2000 a.tcl*
-rw-r--r--  1 aviad    None        36 Jan 27  1994 aa.lst
-rw-r--r--  1 aviad    None       495 Apr 20  1994 aa1.box
-rw-r--r--  1 aviad    None     78973 Apr 20  1994 aa1.mmr
-rw-r--r--  1 aviad    None       265 Apr 20  1994 aa2.box
-rw-r--r--  1 aviad    None        66 Jan 27  1994 aa2.dic
-rw-r--r--  1 aviad    None     10537 Apr 20  1994 aa2.mmr
-rw-r--r--  1 aviad    None       392 Apr 20  1994 aa3.box
-rw-r--r--  1 aviad    None     10647 Apr 20  1994 aa3.mmr
-rw-r--r--  1 aviad    None       265 Apr 20  1994 aa4.box
-rw-r--r--  1 aviad    None     10200 Apr 20  1994 aa4.mmr
-rw-r--r--  1 aviad    None       268 Apr 20  1994 aa5.box
-rw-r--r--  1 aviad    None     16413 Apr 20  1994 aa5.mmr
-rw-r--r--  1 aviad    None        26 Apr 25  1994 aa5.newbox
-rw-r--r--  1 aviad    None       288 Apr 20  1994 aa6.box
-rw-r--r--  1 aviad    None     17486 Apr 20  1994 aa6.mmr
-rw-r--r--  1 aviad    None       339 Nov 27  2000 aggresive.dbg
-rw-r--r--  1 aviad    None     25493 Nov 19  2000 all
-rw-r--r--  1 aviad    None     36760 Nov 13  2000 all.by_street
-rw-r--r--  1 aviad    None      5224 Jan 27  1994 all.lst
-rw-r--r--  1 aviad    None     25493 Nov 19  2000 all.sorted
-rw-r--r--  1 aviad    None     13969 Nov 15  2000 all.tab
-rw-r--r--  1 aviad    None       148 Aug  9  2001 allnames.lst
-rw-r--r--  1 aviad    None       328 Jan 27  1994 allnames.lst~
-rw-r--r--  1 aviad    None     47210 Jan 27  1994 allswiss.dic
-rw-r--r--  1 aviad    None     91562 Aug  9  2001 allswiss.lst
-rw-r--r--  1 aviad    None     78018 Jan 27  1994 allswiss.lst~
-rw-r--r--  1 aviad    None     48040 Nov 15  2000 alpha
-rw-r--r--  1 aviad    None       112 Nov 19  2000 b.awk
-rw-r--r--  1 aviad    None      1245 Nov 27  2000 bad
-rw-r--r--  1 aviad    None      1445 Nov 27  2000 bad.65
-rw-r--r--  1 aviad    None      1293 Nov 27  2000 bad.70
-rw-r--r--  1 aviad    None      1242 Jan 27  1994 bad1
-rw-r--r--  1 aviad    None      1253 Jan 27  1994 bad2
-rw-r--r--  1 aviad    None       539 Jan 27  1994 baseline.dbg
-rw-r--r--  1 aviad    None     13853 Nov 28  1994 black.c
-rw-r--r--  1 aviad    None       193 Nov 19  2000 c.awk
```

```
drwxr-xr-x  1 aviad   None        0 Jan 30  2001 chicago/
-rw-r--r--  1 aviad   None       14 Nov 13  2000 chicago.box
-rw-r--r--  1 aviad   None     9516 Nov 16  2000 chicago.dic
-rw-r--r--  1 aviad   None    48560 Nov 15  2000 chicago.lst
-rw-r--r--  1 aviad   None       24 Jan 27  1994 city.box
-rw-r--r--  1 aviad   None    14135 Jan 27  1994 cityname.lst
-rw-r--r--  1 aviad   None    30783 Nov 19  2000 clean
-rw-r--r--  1 aviad   None    39308 Nov 22  2000 clean.aggresive
-rw-r--r--  1 aviad   None    41390 Nov 16  2000 cmd
-rw-r--r--  1 aviad   None     2060 Jan 27  1994 conf.mat
-rw-r--r--  1 aviad   None     2745 Jan 27  1994 confuse.c
-rw-r--r--  1 aviad   None       55 Nov 22  2000 count.awk
-rw-r--r--  1 aviad   None      732 Jan 27  1994 dat
-rw-r--r--  1 aviad   None       14 Jan 27  1994 dat.box
-rw-r--r--  1 aviad   None      356 Dec 12  2000 dbg
-rw-r--r--  1 aviad   None       27 Dec 12  2000 dbg_huh
-rw-r--r--  1 aviad   None      356 Dec 12  2000 dbg_huh~
-rw-r--r--  1 aviad   None      355 Jan 27  1994 dbg~
-rw-r--r--  1 aviad   None        1 Apr 25  1994 dic.dic
-rw-r--r--  1 aviad   None     4506 Jan 27  1994 digits.tab
-rwxr-xr-x  1 aviad   None       21 Jan 27  1994 disp*
-rw-r--r--  1 aviad   None       30 Aug 24  2000 e.lst
-rw-r--r--  1 aviad   None    35861 Jul 10  1994 faxnames.ioc
-rw-r--r--  1 aviad   None      312 Nov 27  2000 fix.awk
-rw-r--r--  1 aviad   None     7023 Nov 28  1994 fixblack.c
-rw-r--r--  1 aviad   None    46843 Nov 27  2000 fixed
-rw-r--r--  1 aviad   None      533 Jan 27  1994 fixslant.lst
-rw-r--r--  1 aviad   None      532 Jan 27  1994 fld.lst
-rw-r--r--  1 aviad   None     4389 Nov 21  2000 good
-rw-r--r--  1 aviad   None      790 Jan 27  1994 graph.bug
-rw-r--r--  1 aviad   None       83 Jan 27  1994 hard.box
-rw-r--r--  1 aviad   None      186 Jan 27  1994 hard.lst
-rw-r--r--  1 aviad   None       12 Dec 12  2000 hoxie.dic
-rw-r--r--  1 aviad   None        6 Dec 12  2000 hoxie.dic~
-rw-r--r--  1 aviad   None        6 Dec 12  2000 hoyne.dic
-rw-r--r--  1 aviad   None        6 Dec 12  2000 hoyne.dic~
-rw-r--r--  1 aviad   None     1783 Dec 12  2000 hoyne_letters.tab
-rw-r--r--  1 aviad   None     1138 Jan 27  1994 hwr_api.c
-rw-r--r--  1 aviad   None      590 Jan 27  1994 hwr_api.h
-rw-r--r--  1 aviad   None      481 Jan 27  1994 img.lst
-rw-r--r--  1 aviad   None     1061 Jan 27  1994 img1
-rw-r--r--  1 aviad   None     1070 Jan 27  1994 img2
-rw-r--r--  1 aviad   None     1650 Jan 27  1994 img3
-rw-r--r--  1 aviad   None     1670 Jan 27  1994 img_1
-rw-r--r--  1 aviad   None     1682 Jan 27  1994 img_1a
-rw-r--r--  1 aviad   None   535962 Nov 21  2000 imgmatch
-rw-r--r--  1 aviad   None     1130 Nov 21  2000 imgmatch.c
-rw-r--r--  1 aviad   None     2975 Nov 21  2000 imgmatch.o
-rw-r--r--  1 aviad   None   538929 Nov 22  2000 imgsort
-rw-r--r--  1 aviad   None  [redacted]  2000 imgsort.c
-rw-r--r--  1 aviad   None  [redacted]  2000 imgsort.o
-rw-r--r--  1 aviad   None  [redacted]  2000 imgsort0.c
-rw-r--r--  1 aviad   None     1508 Nov 22  2000 k.lst
```

```
-rw-r--r--  1 aviad  None     9352 Jan 27  1994 letters.save
-rw-r--r--  1 aviad  None     9463 Jan 27  1994 letters.tab
-rw-r--r--  1 aviad  None    50332 Jan 30  2001 list
-rw-r--r--  1 aviad  None    84796 Aug  9  2001 log
-rw-r--r--  1 aviad  None    17115 Aug 24  2000 log.base
-rw-r--r--  1 aviad  None    86614 Aug  9  2001 log0
-rw-r--r--  1 aviad  None    89740 Nov 14  2000 m
-rw-r--r--  1 aviad  None    18848 Aug 20  2000 matching
-rw-r--r--  1 aviad  None    22858 Aug 24  2000 matching.c
-rw-r--r--  1 aviad  None    12908 Aug 24  2000 matching.o
-rw-r--r--  1 aviad  None     1408 Nov  9  2000 meiri.1
-rw-r--r--  1 aviad  None     1768 Nov  9  2000 meiri.2
-rw-r--r--  1 aviad  None     2314 Jan 27  1994 names.1
-rw-r--r--  1 aviad  None     2758 Jan 27  1994 names.2
-rw-r--r--  1 aviad  None      514 Apr 20  1994 names.box
-rw-r--r--  1 aviad  None       36 Jan 27  1994 names.lst
-rw-r--r--  1 aviad  None     6921 Aug  9  2001 names.mmr
-rw-r--r--  1 aviad  None      532 Aug 24  2000 new.dbg
-rw-r--r--  1 aviad  None     2171 Nov 15  2000 new.dic
-rw-r--r--  1 aviad  None    53829 Nov 27  2000 new.out
-rw-r--r--  1 aviad  None     1508 Nov 22  2000 new_K.lst
-rw-r--r--  1 aviad  None     3061 Nov 22  2000 new_U.lst
-rw-r--r--  1 aviad  None    23148 Nov 22  2000 new_Unit.lst
-rw-r--r--  1 aviad  None     1508 Nov 22  2000 new_k.lst
-rw-r--r--  1 aviad  None     3061 Nov 22  2000 new_u.lst
-rwxr-xr-x  1 aviad  None      116 Jan 27  1994 ocr*
-rw-r--r--  1 aviad  None     7616 Nov 28  2000 others
-rw-r--r--  1 aviad  None    46899 Nov 28  2000 output
-rw-r--r--  1 aviad  None     3431 Jan 27  1994 profile.c
-rw-r--r--  1 aviad  None      583 Apr 25  1994 profile.dbg
-rw-r--r--  1 aviad  None    24863 Aug 24  2000 quick.c
-rw-r--r--  1 aviad  None    17952 Aug 24  2000 quick.o
-rw-r--r--  1 aviad  None    21947 Jan 27  1994 quick1.c
-rw-r--r--  1 aviad  None     8908 Nov 28  2000 r
-rw-r--r--  1 aviad  None       36 Jan 27  1994 r9.img
-rw-r--r--  1 aviad  None      491 Jan 27  1994 refswiss.dic
-rw-r--r--  1 aviad  None     4896 Jan 27  1994 refswiss.lst
-rw-r--r--  1 aviad  None     3496 Nov 28  2000 rejected
-rwxr-xr-x  1 aviad  None      206 Nov 27  2000 run.sh*
-rwxr-xr-x  1 aviad  None      153 Nov 22  2000 run2.sh*
-rw-r--r--  1 aviad  None    19618 Nov 30  2000 s
-rwxr-xr-x  1 aviad  None      435 Dec  3  2000 s.awk*
-rw-r--r--  1 aviad  None     6869 Jan 27  1994 s_swiss.dic
-rw-r--r--  1 aviad  None      145 Nov 28  2000 select.awk
-rw-r--r--  1 aviad  None   323032 Nov 28  2000 selected
-rw-r--r--  1 aviad  None       69 Mar 11  2004 sh.sh
-rw-r--r--  1 aviad  None       31 Mar 11  2004 sh.sh~
-rw-r--r--  1 aviad  None      424 Jan 27  1994 short.dic
-rw-r--r--  1 aviad  None      460 Feb  8  2001 short.list
-rw-r--r--  1 aviad  None    10647 Aug 24  2000 shortswiss.lst
-rw-r--r--  1 aviad  None    32350 Jan 27  1994 skew.ind
-rw-r--r--  1 aviad  None      579 Jan 27  1994 skew.lst
-rw-r--r--  1 aviad  None     2573 Dec 12  2000 some_letters.tab
```

```
-rw-r--r--  1 aviad  None     9463 Dec 12  2000 some_letters.tab~
-rw-r--r--  1 aviad  None    39005 Nov 16  2000 sorted
-rw-r--r--  1 aviad  None      156 Nov 30  2000 street_side.awk
-rw-r--r--  1 aviad  None    68260 Nov 13  2000 streets
-rw-r--r--  1 aviad  None       31 Nov 30  2000 sum.awk
-rw-r--r--  1 aviad  None    10291 Nov 20  2000 sure
-rw-r--r--  1 aviad  None    10291 Nov 21  2000 sure.sorted
-rw-r--r--  1 aviad  None    47014 Jan 27  1994 swiss.dic
-rw-r--r--  1 aviad  None      588 Jan 27  1994 swiss.lst
-rw-r--r--  1 aviad  None      343 Jan 27  1994 t.lst
-rw-r--r--  1 aviad  None       96 Jan 27  1994 t240.box
-rw-r--r--  1 aviad  None      248 Jan 27  1994 t240.lst
-rw-r--r--  1 aviad  None        6 Nov 21  2000 temp.dic
-rw-r--r--  1 aviad  None    49516 Nov 16  2000 tmp
-rw-r--r--  1 aviad  None       92 Jan 27  1994 tnt.box
-rw-r--r--  1 aviad  None      561 Jan 27  1994 tnt.lst
-rw-r--r--  1 aviad  None     3961 Nov 22  2000 u
-rw-r--r--  1 aviad  None     3061 Nov 22  2000 u.lst
-rw-r--r--  1 aviad  None     2408 Nov 22  2000 u0
-rw-r--r--  1 aviad  None     1191 Nov 22  2000 u1
-rw-r--r--  1 aviad  None     1371 Nov 28  2000 vote.awk
-rw-r--r--  1 aviad  None    16081 Nov 27  2000 voted
-rw-r--r--  1 aviad  None    17568 Nov 27  2000 voted.62
-rw-r--r--  1 aviad  None    17071 Nov 27  2000 voted.65
-rw-r--r--  1 aviad  None    16499 Nov 27  2000 voted.70
-rw-r--r--  1 aviad  None    16223 Nov 28  2000 voted2
-rw-r--r--  1 aviad  None     7963 Nov 15  2000 with_digits
-rw-r--r--  1 aviad  None     9532 Nov 15  2000 wordlist
-rw-r--r--  1 aviad  None   542482 Nov 21  2000 wordreco.imageapp
-rw-r--r--  1 aviad  None     9177 Nov 13  2000 wordreco.o
-rw-r--r--  1 aviad  None        0 Mar 11  2004 wrong.lst
-rw-r--r--  1 aviad  None    47634 Aug  9  2001 x

-rwxr-xr-x  1 aviad  None    16605 [redacted] ChicagoKeyIn.tcl*
-rwxr-xr-x  1 aviad  None    16603 [redacted] ChicagoKeyIn.tcl*~
-rwxr-xr-x  1 aviad  None      875 Nov 14  2000 CompRes.tcl*
-rwxr-xr-x  1 aviad  None      744 Nov 14  2000 CompRes.tcl*~
-rw-r--r--  1 aviad  None     1812 Nov 19  2000 Compare.res
-rwxr-xr-x  1 aviad  None    41483 Nov  8  2000 DPAGKeyIn.tcl*
-rw-r--r--  1 aviad  None    29124 Nov 29  2000 Direction.lst
-rw-r--r--  1 aviad  None    12630 Nov 19  2000 KeyInData.aviad
-rw-r--r--  1 aviad  None    23148 Nov 23  2000 KeyInData.lst
-rw-r--r--  1 aviad  None    21828 Nov 13  2000 KeyInData.lst.aviad
-rw-r--r--  1 aviad  None     3400 Nov 13  2000 KeyInData.lst.aviad100
-rw-r--r--  1 aviad  None      102 Nov 16  2000 KeyInData.lst.bug
-rw-r--r--  1 aviad  None    23148 Nov 23  2000 KeyInData.lst.demo
-rw-r--r--  1 aviad  None    44185 Nov 15  2000 KeyInData.lst.fix
-rw-r--r--  1 aviad  None    43956 Nov 13  2000 KeyInData.lst.old
-rw-r--r--  1 aviad  None    43973 Nov 13  2000 KeyInData.lst.org
-rw-r--r--  1 aviad  None    12728 Nov 23  2000 KeyInData.res
-rw-r--r--  1 aviad  None    12540 Nov 19  2000 KeyInData.res.aviad_lst
-rw-r--r--  1 aviad  None     8671 Nov  9  2000 chicago_ocr.c
```

```
-rw-r--r--  1 aviad  None   1864433 Jan  4  2001 etwish
-rw-r--r--  1 aviad  None     18525 Nov  8  2000 img.lst
-rw-r--r--  1 aviad  None       632 Nov 14  2000 kk
-rw-r--r--  1 aviad  None     43130 Nov  2  2000 take_img.final
-rw-r--r--  1 aviad  None     17778 Nov 14  2000 truth.str
```